



Devang Patel Institute of Advance Technology and Research

(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./Mrs. Prabin Bhagchandani

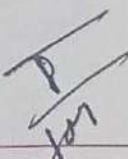
of 6CE1 *Class,*

ID. No. 22DCE006 *has satisfactorily completed*

his/ her term work in CE366 - Cloud Computing *for*

the ending in April 2024/2025

Date : 11/04/25


Sign. of Faculty


Head of Department

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

List of Experiments

Subject: Cloud Computing (CE366) (6th Semester)

Sr. No	Name of Experiment	Hours
1.	<p>Scenario: Frank and Martha are a married team who own and operate a small café business that sells desserts and coffee. Their daughter, Sofía, works at the café. Sofía is pursuing a degree in cloud computing at a local university in the evenings and on the weekends. She has Python development skills and is learning more about how to develop solutions in the cloud. Sofía is eager to start developing a web presence for the café. She thinks that before she starts coding, it would be a good idea to decide on a development environment for developing and running her code. She decides to explore at least two options that are available on AWS.</p> <p>Lab overview and objectives: In this lab, you will take on the role of Sofía. You will connect to an AWS CloudShell environment and explore its capabilities. You will also launch an instance of AWS Cloud9, connect to it, and explore the layout and functionality of its integrated development environment (IDE). In addition, you will use Amazon CodeWhisperer inside the AWS Cloud9 IDE to generate a Python script.</p> <p>After completing this lab, you should be able to do the following:</p> <ul style="list-style-type: none"> • Connect to AWS CloudShell and run AWS Command Line Interface (AWS CLI) commands and AWS SDK code from it. • Create an AWS Cloud9 development environment and connect to the browser-based IDE. • Copy files to and from Amazon Simple Storage Service (Amazon S3), CloudShell, and AWS Cloud9. • Install the AWS SDK for Python (Boto3) on an AWS Cloud9 instance. • Use the AWS Cloud9 development environment to create files and run code files. • Use Amazon CodeWhisperer in AWS Cloud9 to generate code to interact with AWS services. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	02
2.	<p>Scenario: Sofia is eager to start building a website for the café. She has some Python development skills and she's learning more about how to develop solutions on the cloud. Nikhil is a secondary school student who also works at the café. He has some skills in graphic design and a strong interest in learning about cloud computing. The café has a single location in a large city. It mostly gains new customers when someone walks by, notices the café, and decides to try it. The café has a reputation for high-quality desserts and coffees, but their reputation is limited to people who have visited, or who have heard about them from their customers. Sofia suggests to Frank and Martha (her parents and the owners of the café) that</p>	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<p>they should expand community awareness of what the café has to offer. The café doesn't have a web presence, and it doesn't currently use any cloud computing services. However, that situation is about to change. In this lab, you will play the role of Sofía and take the first steps needed create a website for the café.</p> <p>Lab overview and objectives: In this lab, you use Amazon Simple Storage Service (Amazon S3) to host a static website. You will also implement architectural best practices to protect and manage your data.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a static website by using Amazon S3. • Apply a bucket policy on an S3 bucket to configure customized data protection. • Upload objects to an S3 bucket by using the AWS SDK for Python (Boto3). • Configure the website that is hosted on Amazon S3 to be accessible only from a specific IP address range, and test the configuration. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
3.	<p>Scenario: The café website is up and running, and the café staff noticed a significant increase in new customer visits. Multiple customers also mentioned that it would be helpful if the website had an up-to-date menu. They could then use the menu to check the availability of food items before going to the café. Frank and Martha ask Sofía to explore whether she can implement this feature for customers. Sofía is feeling more confident in her coding skills and has also been learning about different ways to store information in AWS. She knows that before they can dynamically update data on the website, she must first choose a data storage service to hold the data. She also needs to learn how to manage table data, load the product records, and create scripts to retrieve information from the data platform. A business request from the café: Store menu information in the cloud. Frank and Martha mentioned to Sofía that they want the website to dynamically update its menu information. To prepare for this new functionality, Sofía decides to store this information in DynamoDB. Café staff must be able to retrieve information from the table. Sofía decides to create one script that retrieves all inventory items from the table and another script (as a proof of concept) that uses a product name to retrieve a single record. For this first challenge, you take on the role of Sofía. You use the AWS CLI and the SDK for Python to configure and create a DynamoDB table, load records into the table, and extract data from the table.</p> <p>Lab overview and objectives: In this lab, you use Amazon DynamoDB to store and manage menu information. Using databases, such as DynamoDB, simplifies data management because you can easily query, sort, edit, and index data. You will use both the AWS Command Line Interface (AWS CLI) and the AWS SDK for Python (Boto3) to work with DynamoDB. In upcoming labs, you will use application programming interface (API) calls from the café website to dynamically retrieve and update data that's stored in a DynamoDB table.</p> <p>After completing this lab, you should be able to:</p>	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<ul style="list-style-type: none"> • Create a new DynamoDB table. • Add data to the table. • Modify table items based on conditions. • Query the table. • Add a global secondary index to the table. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
4.	<p>Scenario: In the previous lab, you took on the role of Sofía to build a web application for the café. As part of this process, you created an Amazon DynamoDB table that was named FoodProducts, where you stored information about café menu items. You then loaded data that was formatted in JavaScript Object Notation (JSON) into the database table. In the previous lab you also configured code that used the AWS SDK for Python (Boto3) to:</p> <ul style="list-style-type: none"> • Scan a DynamoDB table to retrieve product details. • Return a single item by product name using get-item as a proof of concept. • Create a Global Secondary Index (GSI) called special_GSI that you could use to filter out menu items that are on offer and not out of stock. <p>In this lab, you will continue to play the role of Sofía. You will use Amazon API Gateway to configure mock data endpoints. There are three that you will create:</p> <ul style="list-style-type: none"> • [GET] /products (which will eventually invoke a DynamoDB table scan). • [GET] /products/on_offer (which will eventually invoke a DynamoDB index scan and filter). • [POST] /create_report (which will eventually trigger a batch process that will send out a report). <p>Then in the lab that follows this one, you will replace the mock endpoints with real endpoints, so that the web application can connect to the DynamoDB backend.</p> <p>Lab overview and objectives: In this lab, you will create a REST application programming interface (API) by using Amazon API Gateway.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create simple mock endpoints for REST APIs and use them in your website. • Enable Cross-Origin Resource Sharing (CORS). <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	02
5.	<p>Scenario: The café is eager to launch a dynamic version of their website so that the website can access data stored in a database. Sofía has been making steady progress toward this goal. In a previous lab, you played the role of Sofía and created a DynamoDB database. The database table contains café menu details, and an index holds menu items that are flagged as specials. Then, in another lab, you created an API to add the ability for the website to receive mock</p>	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<p>data through REST API calls. In this lab, you will again play the role of Sofia. You will replace the mock endpoints with functional endpoints so that the web application can connect to the database. You will use Lambda to bridge the connection between the GET APIs and the data stored in DynamoDB. Finally, for the POST API call, Lambda will return an updated acknowledgment message.</p> <p>Lab overview and objectives: In this lab, you will use the AWS SDK for Python (boto3) to create AWS Lambda functions. Calls to the REST API that you created in the earlier Amazon API Gateway lab will initiate the functions. One of the Lambda functions will perform either an Amazon DynamoDB database table scan or an index scan. Another Lambda function will return a standard acknowledgment message that you will enhance later in a lab where implement Amazon Cognito.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a Lambda function that queries a DynamoDB database table. • Grant sufficient permissions to a Lambda function so that it can read data from DynamoDB. • Configure REST API methods to invoke Lambda functions using Amazon API Gateway. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
6.	<p>Scenario: The café owners have noticed how popular their gourmet coffee offerings have become. Customers cannot seem to get enough of their cappuccinos and lattes. Meanwhile, the café owners have been challenged to consistently source the highest quality coffee beans. Recently, the owners learned that one of their favorite coffee suppliers wants to sell her company. Frank and Martha jumped at the opportunity to buy the company. The acquired coffee supplier runs an inventory tracking application on an AWS account. Sofia has been tasked to understand how the application works and then create a plan to integrate the application into the café's existing application infrastructure. In this lab, you will again play the role of Sofia, and you will work to migrate the application to run on containers. By the end of this lab, you will have migrated the application and the backend database to run as Docker containers. You will register these two containers in Amazon ECR to make them available to deploy as needed.</p> <p>Lab overview and objectives: In this lab, you will migrate a web application to run on Docker containers. The application is installed directly on the guest operating systems (OSs) of two Amazon Elastic Compute Cloud (Amazon EC2) instances. You will migrate the application to run on Docker containers.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a Dockerfile. • Create a Docker image by using a Dockerfile. • Run a container from a Docker image. • Interact with and administer your containers. 	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<ul style="list-style-type: none"> • Create an Amazon Elastic Container Registry (Amazon ECR) repository. • Authenticate the Docker client to Amazon ECR. • Push a Docker image to Amazon ECR. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
7.	<p>Scenario: Sofia has containerized the coffee suppliers application. Now the café has asked if she can reduce the required manual application maintenance and plan for the future scalability of the environment. She knows from her studies that Amazon Web Services (AWS) provides several managed services. Managed services can help to reduce the burden of deploying and managing applications. After more research, she has decided to deploy the suppliers application website using Elastic Beanstalk. However, Sofia has discovered that scaling a relational database using containers is not recommended because relational databases are stateful. Relational databases require reliable communication between database hosts and storage. This is difficult to accomplish using dynamic containers. Therefore, Sofia has decided to use Aurora Serverless as the data platform. Sofia will retire the container-based MySQL database and load the required user, tables, and data into an Aurora Serverless database. Aurora Serverless was designed to seamlessly and safely scale databases as transaction loads increase. As a bonus, when the database isn't being used, Aurora Serverless automatically scales down, which will save money for the café. In this lab, you will again play the role of Sofia, and you will deploy the suppliers application using managed services.</p> <p>Lab overview and objectives: In a previous lab, you migrated an application that ran on Amazon Elastic Compute Cloud (Amazon EC2) instances to run on Docker containers. In this lab, you will deploy the application using two managed cloud services. You will deploy the database tier using Amazon Aurora Serverless and the web tier using AWS Elastic Beanstalk.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a new Amazon Relational Database Service (Amazon RDS) instance using the AWS Management Console. • Launch a Docker container on AWS Cloud9 using an image pulled from Amazon Elastic Container Registry (Amazon ECR). • Configure and test the containerized application connection to Aurora Serverless. • Use the Amazon RDS query editor to create database objects and load data. • Launch the default Elastic Beanstalk application. • Update the Elastic Beanstalk application to run your node application and communicate with Amazon RDS. • Configure an Amazon API Gateway endpoint to forward calls to the Elastic Beanstalk URL. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	02
8.	<p>Scenario: Frank and Martha are excited that the bean inventory from the coffee suppliers application is integrated into the café website. Unfortunately, they are getting occasional negative</p>	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<p>feedback from customers who say that the bean inventory information takes too long to display on the page. Frank and Martha have asked Sofía if she can have the site display the information more quickly. Luckily, one of the café's customers, Olivia, is an AWS consultant and has a lot of database experience. She has offered to help with the website. Olivia proposed adding database caching to improve the speed of the database query. This way, frequently used data would be stored in memory instead of having to be retrieved from the database, which could also require information to be read from database storage. Olivia explained that you could cache data locally on the Amazon Elastic Compute Cloud (Amazon EC2) instance that is hosting the application containers. However, that architecture would not provide fault tolerance. This strategy would also require more administration because you would need to install and maintain the caching software yourself. Olivia recommended using the Amazon ElastiCache managed service. ElastiCache is similar to Aurora Serverless in that it simplifies the process to deploy and maintain a cache cluster. In this lab, you will act as Olivia to work through the process to deploy and test the ElastiCache cluster. You will also play the role of Nikhil to update and redeploy the coffee suppliers application to use caching on the website.</p> <p>Lab overview and objectives: In this lab, you will deploy an Amazon ElastiCache cluster. You will also test synchronizing the cache with the Amazon Aurora Serverless database by using Python scripts. Finally, you will update the coffee suppliers application with new Node.js code that will use data caching.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a new ElastiCache for Memcached cluster. • Query the ElastiCache for Memcached cluster by using Python and the pymemcache client. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
9.	<p>Scenario: Sofía is pleased with how the café website development project is coming along. She has developed the core serverless application that displays menu items on the website. She also integrated the coffee suppliers web application into the main site and is using Amazon ElastiCache features for the suppliers part of the site. However, she knows that some essential features are still missing. One feature is that the website still runs on HTTP and does not yet support HTTPS. Sofía also wants to ensure that the website will load quickly for users globally. She knows that AWS has many Regions and Availability Zones, but they also have edge locations, which are even closer to users around the globe. She decides to host the café website on a proper content delivery network (CDN), and she has opted to use the CloudFront service. In this lab, you will again play the role of Sofía to continue to develop the café's web application.</p> <p>Lab overview and objectives: In this lab, you will create an Amazon CloudFront distribution to reduce network latency for café website users and deliver the website securely over HTTPS. You will also secure access to the website and REST API endpoints using AWS WAF, which is a service that provides a web application firewall. Finally, you will configure a CloudFront function on the website and adjust the cached file expiration time on the website content.</p>	04

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a CloudFront distribution to cache Amazon Simple Storage Service (Amazon S3) objects. • Configure a website hosted on Amazon S3 to be available through HTTPS using CloudFront. • Secure access to the CloudFront distribution based on the network origin of the request. • Secure a REST API endpoint based on the network origin of the request using AWS WAF. • Configure a CloudFront function to affect website behavior from the edge. • Adjust max-age caching settings on a CloudFront distribution. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
10.	<p>Scenario: Customers love being able to buy coffee beans from the café, but maintaining the bean inventory is time-consuming. Café employees must call each coffee supplier and then use the coffee suppliers web application to manually update each record. Frank has asked Sofía if she can find a better way to keep the coffee inventory current. He would like café staff to spend more time helping customers and less time doing data entry. Sofía has researched setting up a messaging system to automatically receive and process inventory updates. Mateo, a café regular and AWS consultant, suggested using an SNS topic to receive messages from suppliers, and an SQS queue to store the messages until the application is ready to process them. This way, the café application won't lose any messages if the application or database happens to be unavailable. He also highly recommended that she create a dead-letter queue to handle messages that cannot be processed. Mateo also explained that suppliers will need to run a script called a producer to publish their updates to the SNS topic. The coffee suppliers application code will need to include a consumer to poll and retrieve messages from the SQS queue. In this lab, you will again play the role of Sofía as you develop the automated inventory processing for the café's coffee suppliers application.</p> <p>Lab overview and objectives: In this lab, you will use Amazon Simple Queue Service (Amazon SQS) and Amazon Simple Notification Service (Amazon SNS) to set up a system to receive, queue, and send data for an application to process. You will use a Python publisher to send messages to a notification topic. You will also review the Node.js consumer that a web application will use to retrieve and process the data from a queue.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Configure SNS topics and SQS queues to support programmatic receipt of messages. • Develop an Amazon SNS publisher to send messages to an SNS topic. • Develop an Amazon SQS consumer to read messages from an SQS queue. 	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	Reference: https://awsacademy.instructure.com/courses/85538	
11.	<p>Scenario: Thanks to Sofia, the coffee suppliers inventory now automatically updates on the café website. But the work is not finished! Frank asks if it would be possible to log in to the website to request a report with the latest inventory information. Yesterday, Mateo, who is an AWS consultant and Sofia's friend, came into the café for a macchiato, his favorite espresso drink. While he was enjoying his beverage, Sofia told him about how she needs to build a reporting mechanism for Frank. After discussing the high-level business requirements, Mateo suggested that she use AWS Step Functions to coordinate the steps to generate the report. He described how Step Functions can help a developer to automate business processes by creating workflows, and providing parallelization and service integrations, among other features. In this lab, Sofia will build the functionality to create and deliver the report. Then, in the next lab, she will improve the design further and implement authentication on the website to limit who can request and access reports.</p> <p>Lab overview and objectives: In this lab, you will use AWS Step Functions to coordinate the actions necessary to generate and deliver a report upon request. The report will contain data from a database.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create an asynchronous state machine by using Step Functions. • Configure an Amazon Simple Notification Service (Amazon SNS) topic to deliver email alerts. • Configure AWS Lambda functions to be invoked from a Step Functions state machine. • Use a parallel state flow object in the design of a Step Functions state machine. • Invoke a state machine to start when a REST API endpoint is invoked. • Generate a presigned URL for an object stored in an Amazon Simple Storage Service (Amazon S3) bucket. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	02
12.	<p>Scenario: Frank wants to be able to log in to the café website and request an inventory report to be sent to his email address (on his phone). He wants to be able to see the latest coffee bean inventory data quickly. In this lab, you will play the role of Sofia to implement this technical feature request. In the previous lab, Sofia created a Step Functions state machine that, once invoked, can generate the report that Frank wants. However, the state machine can currently only be invoked by using the test feature in the Step Functions console or by running a curl command to invoke the create_report REST API endpoint. In this lab, Sofia will use Amazon Cognito to integrate an authentication mechanism into the website. Frank will be able to log in to the website to confirm his identity before he requests the report. Then, she will connect the REST API endpoint to the café website so that he can make his report request directly from the site. The API request will include the ID token that is retrieved as part of the authentication process, and Amazon Cognito will be used to validate</p>	02

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING (FTE)
Chandubhai S. Patel Institute of Technology (CSPIT) &
Devang Patel Institute of Advance Technology and Research (DEPSTAR)
ACADEMIC YEAR: 2024-25

	<p>the token. Then, the Step Functions state machine will be invoked, which will then invoke the AWS Lambda functions that generate the report.</p> <p>Lab overview and objectives: In this lab, you will work as Sofía to integrate Amazon Cognito into the café website. Frank wants to be able to log in and request a coffee bean inventory report directly from the café website. Amazon Cognito provides an authentication service, which Sofía wants to use for this website enhancement. This usability enhancement will use the state machine that you built in the previous lab using AWS Step Functions.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create an Amazon Cognito user pool. • Create an Amazon Cognito user pool user. • Configure an app client to use Amazon Cognito as an authentication service. • Integrate an Amazon Cognito app client into a website. • Update REST API endpoints that are built with Amazon API Gateway to call Amazon Cognito for authentication purposes. • Configure an API Gateway authorizer. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	
13.	<p>Scenario: Now that the café website is in production, Frank wants a reliable process in place to track code changes and update the site when changes are made. He asked Sofía to find a way to centralize the website code and add version control. He has also asked if it's possible to automatically update the website instead of manually running scripts and uploading files when changes are made. Mateo, a café regular and AWS consultant who specializes in automating repeatable processes, was chatting with Sofía about his work. He mentioned using CodeCommit to collaborate on projects with other developers. Sofía shared that she was researching CodeCommit to centralize the café website's code and asked Mateo for suggestions about automating updates to the site. Mateo suggested using CodePipeline because it easily integrates with both CodeCommit and Amazon Simple Storage Service (Amazon S3).</p> <p>Lab overview and objectives: In this lab, you will create an AWS CodeCommit repository (also known as a repo) and an AWS CodePipeline pipeline. You will configure the pipeline to automatically apply updates to the café website as changes are saved to the repository.</p> <p>After completing this lab, you should be able to:</p> <ul style="list-style-type: none"> • Create a new CodeCommit repository. • Clone and update a CodeCommit repository. • Create a pipeline by using CodePipeline. <p>Reference: https://awsacademy.instructure.com/courses/85538</p>	04

Grades for 22DCE006@charusat.edu.in

 [Print Grades \(javascript:window.print\(\)\)](#)









Arrange By

Due Date

▼

Apply

Name	Due	Submitted	Status	Score	
Lab 2.1: Exploring AWS CloudShell and an IDE Labs		Dec 31, 2024 at 9:58am		100 / 100	
Lab 3.1: Working with Amazon S3 Labs		Jan 27 at 2:16pm		100 / 100	
Lab 5.1: Working with DynamoDB Labs		Feb 1 at 4:54pm		100 / 100	
Lab 6.1: Developing REST APIs with API Gateway Labs		Feb 4 at 3:01pm		100 / 100	
Lab 7.1: Creating Lambda Functions Using the AWS SDK for Python Labs		Feb 11 at 11:28am		100 / 100	
Lab 8.1: Migrating a Web Application to Docker Containers Labs		Feb 18 at 10:43am		100 / 100	
Lab 8.2: Running Containers on a Managed Service Labs		Mar 4 at 11:09am		12.5 / 100	
Lab 9.1: Caching Application Data with ElastiCache Labs		Mar 16 at 3:07pm		100 / 100	
Lab 9.2: Implementing CloudFront for Caching and Application Security Labs		Mar 11 at 10:38am		100 / 100	

Name	Due	Submitted	Status	Score
Lab 10.1: Implementing a Messaging System Using Amazon SNS and Amazon SQS Labs		Mar 21 at 10:27am		100 / 100 
Lab 11.1: Orchestrating Serverless Functions with Step Functions Labs				- / 100
Lab 12.1: Implementing Application Authentication Using Amazon Cognito Labs		Mar 31 at 10:42pm		100 / 100 
Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline Labs				- / 100
Module 2 Knowledge Check Knowledge Checks		Dec 31, 2024 at 10:14am		100 / 100 
Module 3 Knowledge Check Knowledge Checks		Dec 31, 2024 at 10:52am		100 / 100 
Module 4 Knowledge Check Knowledge Checks		Feb 1 at 2:08pm		100 / 100 
Module 5 Knowledge Check Knowledge Checks		Feb 1 at 3:50pm		100 / 100 
Module 6 Knowledge Check Knowledge Checks		Feb 4 at 10:27am		100 / 100 
Module 7 Knowledge Check Knowledge Checks		Feb 11 at 9:54am		100 / 100 
Module 8 Knowledge Check Knowledge Checks		Feb 18 at 11:03am		100 / 100 
Module 9 Knowledge Check Knowledge Checks		Mar 11 at 10:48am		100 / 100 

Name	Due	Submitted	Status	Score
Module 10 Knowledge Check Knowledge Checks		Mar 18 at 10:47am		100 / 100
Module 11 Knowledge Check Knowledge Checks		Mar 31 at 8:58pm		100 / 100
Module 12 Knowledge Check Knowledge Checks		Mar 31 at 9pm		100 / 100
Module 13 Knowledge Check Knowledge Checks		Apr 1 at 9:48am		100 / 100
Assignments			N/A	0.00 / 0.00
Knowledge Checks			100%	1,200.00 / 1,200.00
Labs			92.05%	1,012.50 / 1,100.00
Total			96.2%	2,212.50 / 2,300.00

